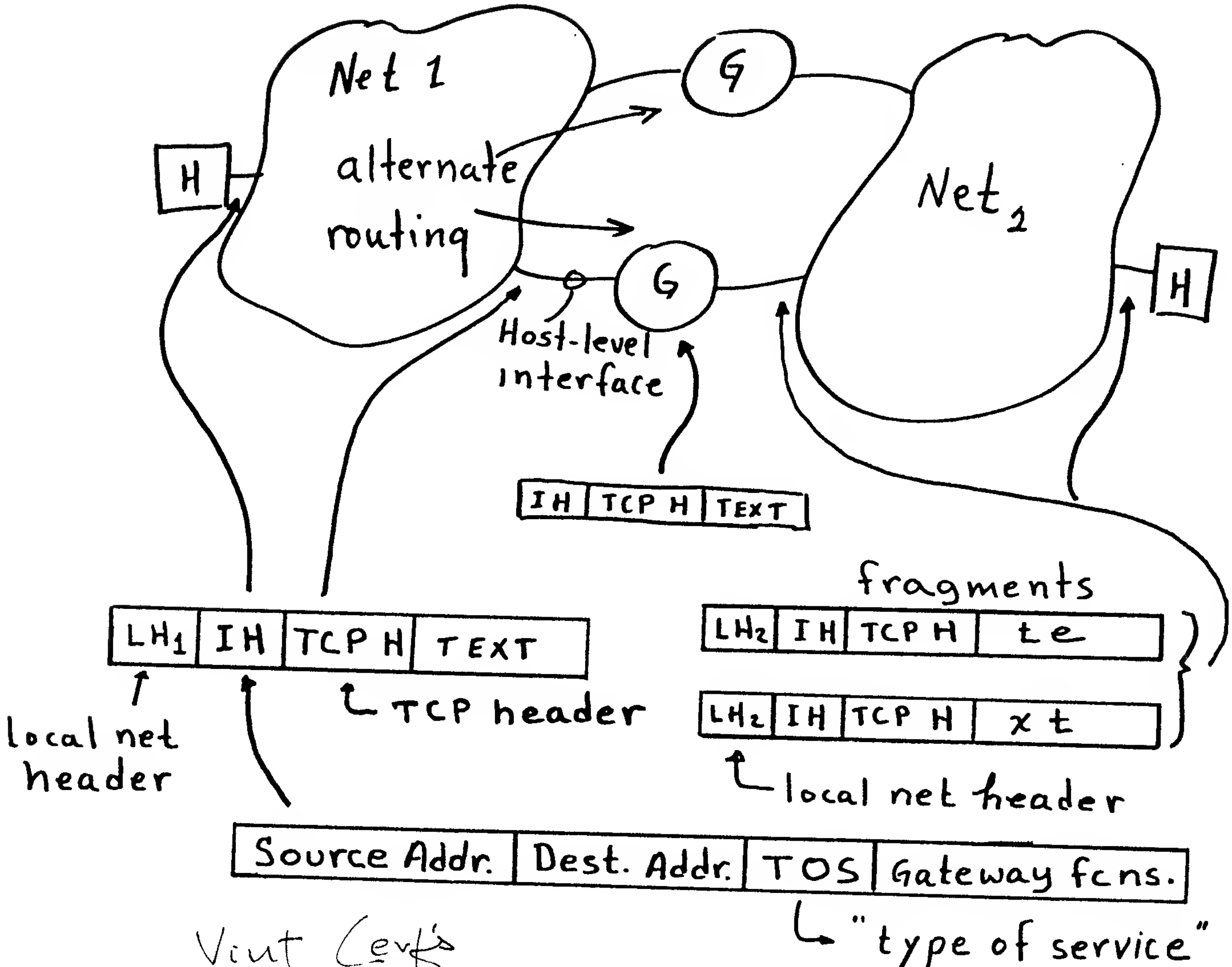
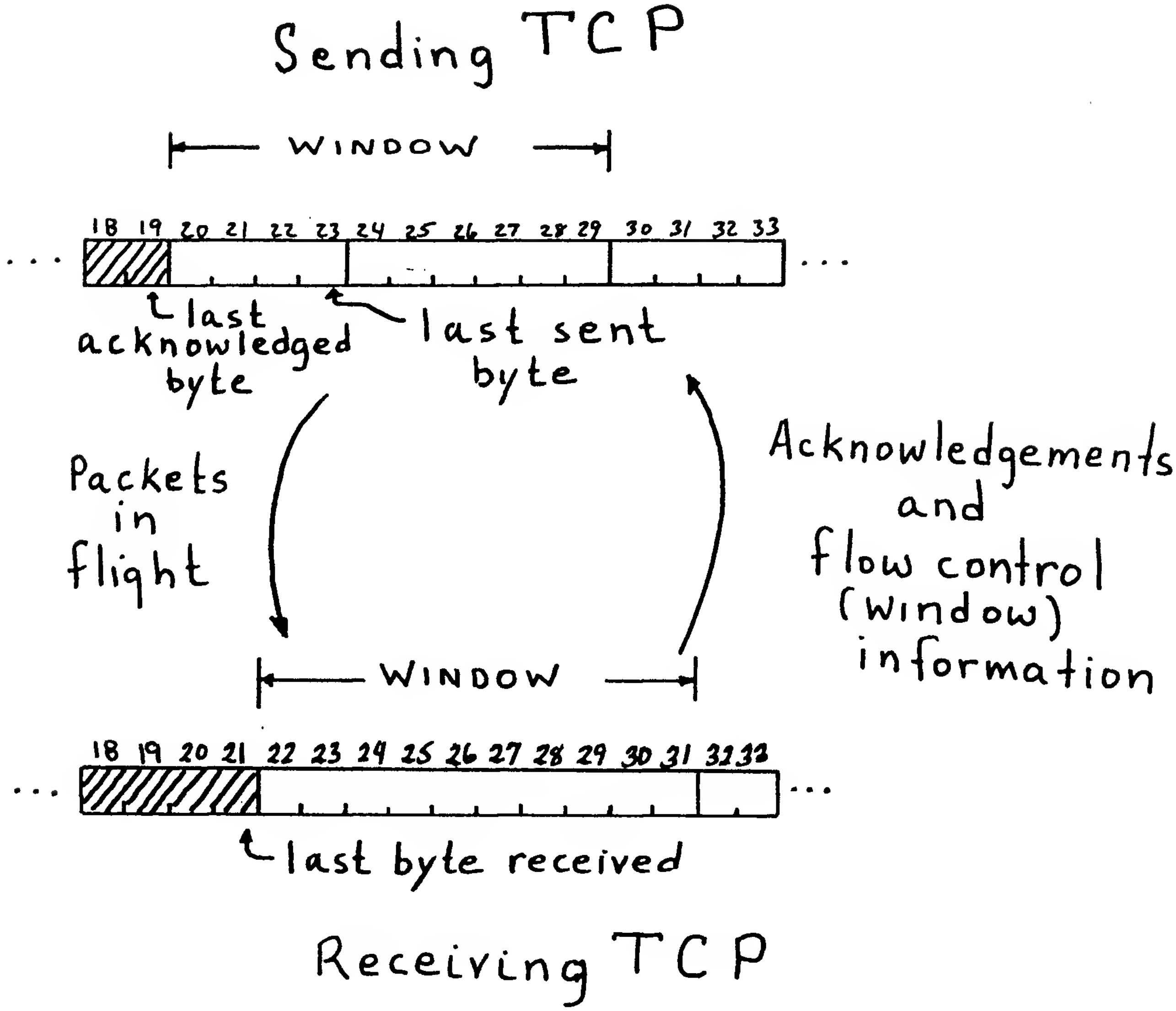
Basic Gateway Concepts



Vint Cerf's
early diagrams
of Internetworking

Flow control



XEROX

EXHIBIT A

ETHER!

Xerox Sensitive

MEMO

MAY 22, 1973

TO: ALTO ALOHA DISTRIBUTION

FROM: BOB METCALFE

SUBJECT: ETHER ACQUISITION

HERE IS MORE ROUGH STUFF ON THE ALTO ALOHA NETWORK.

I PROPOSE WE STOP CALLING THIS THING "THE ALTO ALOHA NETWORK". FIRST, BECAUSE IT SHOULD SUPPORT ANY NUMBER OF DIFFERENT KINDS OF STATION -- SAY, NOVA, PDP-11, ... SECOND, BECAUSE THE ORGANIZATION IS BEGINNING TO LOOK VERY MUCH MORE BEAUTIFUL THAN THE ALOHA RADIO NETWORK -- TO USE CHARLES'S "BEAUTIFUL".

MAYBE: "THE ETHER NETWORK". SUGGESTIONS?

LAZY SUZAN
BULLETIN BOARD
PARLEY
PARLIAMENTARY PROCEDURE

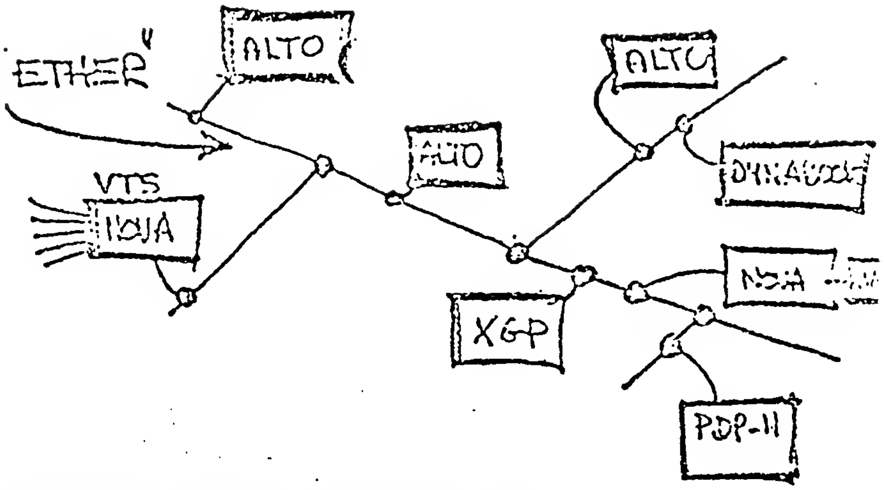
I HOPE TO BE SIMULATING SOON. HELP? INPUTS?

I HOPE YOU WILL NOT BE OFFENDED BY MY ATTEMPTS TO MAKE THIS THINKING AND DESIGN APPEAR THEORETICAL.

Bob

XEROX

"A CABLE-TREE ETHER"

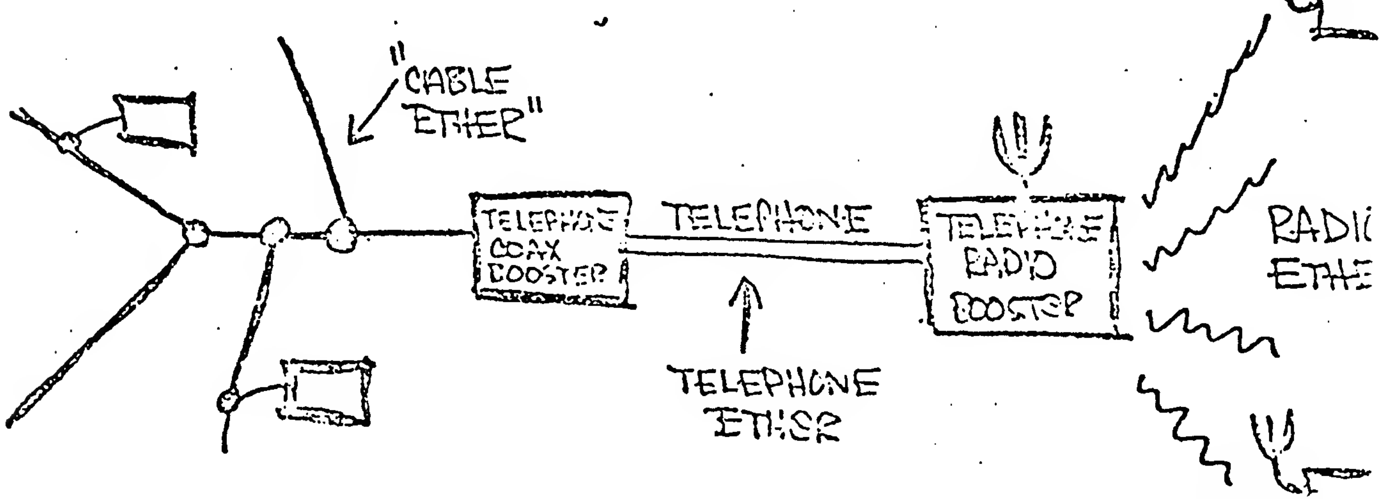


THE ETHER NETWORK

WE PLAN TO BUILD A SO-CALLED BROADCAST COMPUTER COMMUNICATION NETWORK, NOT UNLIKE THE ALOHA SYSTEM'S RADIO NETWORK, BUT SPECIFICALLY FOR IN-BUILDING MINICOMPUTER COMMUNICATION. WE THINK IN TERMS OF NOVA'S AND ALTO'S JOINED BY COAXIAL CABLES.

WHILE WE MAY END UP USING COAXIAL CABLE TREES TO CARRY OUR BROADCAST TRANSMISSIONS, IT SEEMS WISE TO TALK IN TERMS OF AN ETHER, RATHER THAN 'THE CABLE', FOR AS LONG AS POSSIBLE. THIS WILL KEEP THINGS GENERAL AND WHO KNOWS WHAT OTHER MEDIA WILL PROVE BETTER THAN CABLE FOR A BROADCAST NETWORK; MAYBE RADIO OR TELEPHONE CIRCUITS, OR POWER WIRING OR FREQUENCY-MULTI-PLEXED CATV, OR MICROWAVE ENVIRONMENTS, OR EVEN COMBINATIONS THEREOF.

THE ESSENTIAL FEATURE OF OUR MEDIUM — THE ETHER — IS THAT IT CARRIES TRANSMISSIONS, PROPAGATES BITS TO ALL STATIONS. WE ARE TO INVESTIGATE THE APPLICABILITY OF ETHER NETWORKS.



ETHER ACQUISITION

HOW DOES A STATION'S TRANSMITTER ACQUIRE THE USE OF THE ETHER FOR A PARTICULAR TRANSMISSION? THERE ARE MANY POSSIBLE WAYS.

THE ALOHA RADIO NETWORK USES WHAT WE CALL "DE FACTO" ETHER ACQUISITION. A STATION DESIRING TO TRANSMIT SIMPLY DOES, IT JUMPS RIGHT ON AND USES THE ETHER. IF THE TRANSMISSION GOES THROUGH, THE ETHER HAS BEEN SUCCESSFULLY ACQUIRED, DE FACTO. IF SOME OTHER TRANSMISSION CONFLICTS, THEN BOTH (ALL) ARE LOST AND ARE RETRIED SOME RANDOM TIME LATER; THE ETHER HAS FAILED TO BE ACQUIRED.

AT LEAST TWO FACTS ABOUT THE ALOHA ETHER AND TRANSCEIVERS SUPPORT THE USE OF DE FACTO ETHER ACQUISITION. FIRST, THE ALOHA ETHER IS VERY BIG, IT TAKES A LONG TIME FOR TRANSMISSIONS TO PROPAGATE; AND SECOND, ALOHA TRANSCEIVERS ARE STRICTLY HALF-DUPLEX, THEY CANNOT DETECT INTERFERENCE WHILE TRANSMITTING. NEITHER OF THESE TWO FACTS IS TRUE OF OUR ETHER OR OUR STATIONS AS THEY ARE ENVISIONED.

XEROX

AND NOW, FOUR AXIOMS:

Axioms?

(1) THE ETHER AXIOM: THE ETHER CARRIES TRANSMISSIONS TO ALL STATIONS.

(2) THE PROXIMITY AXIOM: PROPAGATION TIMES ARE SOMEWHAT SMALL.

(3) THE DETECTION AXIOM: STATIONS CAN DETECT, AT ALL TIMES, TRANSMISSIONS OF OTHER STATIONS, AS THEY PASS, IN ABOUT ONE BIT TIME.

(4) THE DEFERENCE AXIOM: WHILE DETECTING A PASSING TRANSMISSION, NO STATION WILL BEGIN OR CONTINUE ITS OWN TRANSMISSION.

NOT THE LOCAL NETWORK!

THE ETHER AXIOM FREES US FROM CONSIDERING NETWORK ROUTING. THE PROXIMITY AXIOM ALLOWS US TO CONSIDER SOLUTIONS WHICH WOULD BE TOTALLY IMPRACTICAL OTHERWISE -- SAY AS IN ALOHA RADIO. THE DETECTION AXIOM DOES NOT IMPLY THAT CONFLICTS CAN BE AVOIDED; SEPARATED TRANSCEIVERS CAN BEGIN TRANSMISSION ON FREE ETHER ONLY TO DISCOVER LATER THAT THEIR TRANSMISSIONS HAVE COLLIDED ELSEWHERE. THE DEFERENCE AXIOM FOLLOWS FROM NOTHING MORE THAN OUR BASIC INTUITION — MAYBE IT SHOULD BE DISCARDED SOMETIME..

XEROX

AND NOW, A DEFINITION:

A STATION IS SAID TO HAVE ACQUIRED THE ETHER WHEN AND ONLY WHEN IT HAS BEGUN TRANSMITTING A PACKET AND ALL OF THE OTHER STATIONS HAVE DETECTED THE TRANSMISSION AND ARE DEFERRING TO IT.

AFTER ACQUIRING THE ETHER, A STATION IS SAID TO HOLD THE ETHER AS LONG AS IT CONTINUES TRANSMITTING.

THE DEFERENCE AXIOM IMPLIES THAT ONCE A STATION HAS ACQUIRED THE ETHER, IT CAN HOLD THE ETHER AS LONG AS IT WANTS, USING IT WITHOUT CONFLICT FOR THE DURATION OF ITS TRANSMISSION. A STATION VIOLATING THE DEFERENCE AXIOM COULD, OF COURSE, BREAK A HOLD ON THE ETHER AND ACQUIRE IT, BUT FOR THE MOMENT WE DISALLOW THIS BEHAVIOR.

IF THE ETHER IS TO BE SHARED IN SOME REASONABLE WAY, THEN FURTHER AGREEMENTS WILL BE REQUIRED TO REGULATE THE MAXIMUM HOLDING TIME. BUT THIS COMES LATER.

XEROX

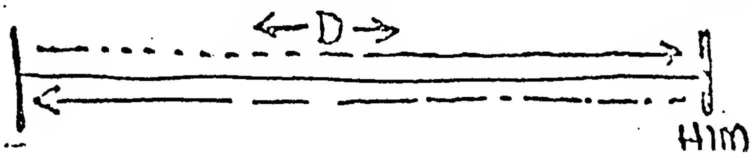
AND NOW, ANOTHER SO-CALLED AXIOM:

(5) THE DIAMETER AXIOM: FOR ANY GIVEN ETHER NETWORK, THERE EXISTS A DIAMETER D, THE PROPAGATION DELAY BETWEEN MOST DISTANT STATIONS, THE MAXIMUM TIME FROM START OF TRANSMISSION TO DETECTION OF TRANSMISSION BY A DISTANT STATION.

BY THE PROXIMITY AXIOM, D IS "SOMEWHAT" SMALL.

AND NOW A FACT:

HOW LONG AFTER BEGINNING TRANSMISSION MUST I DETECT NO CONFLICT BEFORE I CAN BE CERTAIN THAT I HAVE ACQUIRED THE ETHER? THE ANSWER: 2D, ONE ROUND TRIP. SAY THAT THERE IS THIS STATION AT THE FAR END OF THE ETHER, D SECONDS AWAY. AFTER I START TRANSMISSION ON THE OPEN ETHER, IT CAN BE D SECONDS BEFORE HE KNOWS ABOUT IT. BUT IF JUST BEFORE MY TRANSMISSION REACHES HIM HE DECIDES TO TRANSMIT HIMSELF, THEN IT WILL BE D MORE SECONDS BEFORE I FIND OUT ABOUT IT -- IT CAN BE 2D SECONDS BEFORE I SENSE CONFLICT AND THEREFORE FAILURE TO ACQUIRE. HE WILL HAVE SENT A BIT OR TWO BEFORE DETECTING MY TRANSMISSION AND WILL DEFER, BUT IT'S TOO LATE. HIS BRIEF TRANSMISSION WILL CAUSE ME TO LET GO OF THE ETHER ACCORDING TO THE AXIOM OF DEFERENCE. IT TAKES 2D SECONDS OF ETHER TIME TO ACQUIRE.

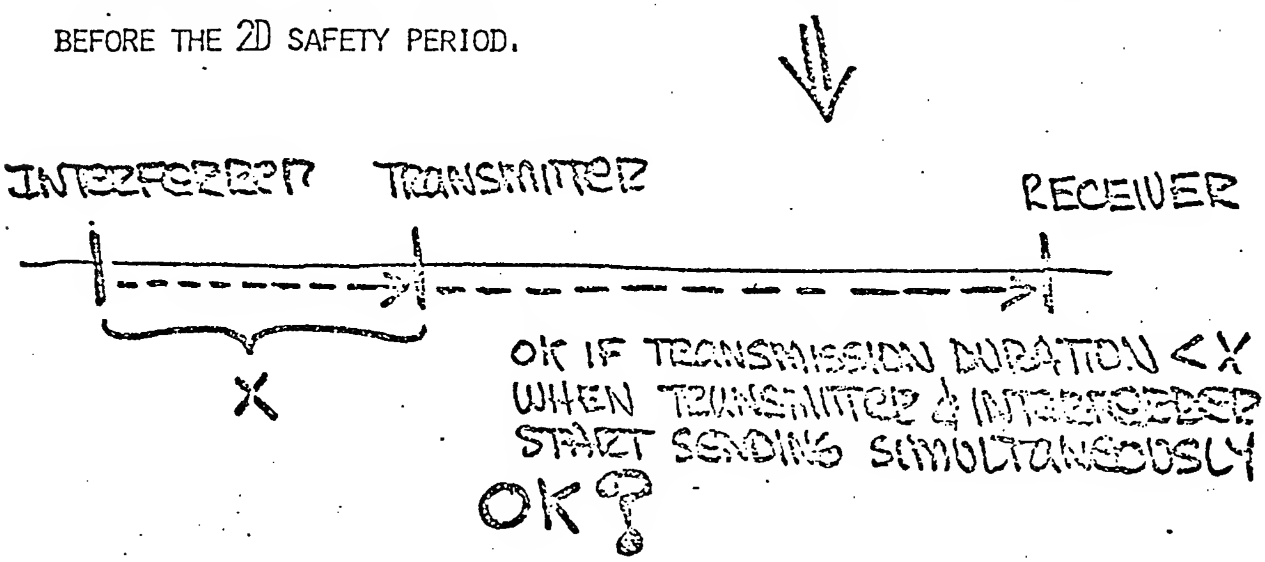


XEROX

DEFINITION: A TRANSMISSION IS SAID TO BE CONFLICT-FREE WITH RESPECT TO ITS TRANSMITTER AND A SPECIFIED RECEIVER (DISREGARDING ETHER NOISE) IF AND ONLY IF THE TRANSMISSION PLACED ON THE ETHER BY THE TRANSMITTER IS LATER CORRECTLY RECEIVED (I.E., WITHOUT INTERFERENCE) AT THE RECEIVER.

FACT: A TRANSMISSION OF ANY LENGTH D (EVEN LESS THAN D) CAN BE DETERMINED TO BE CONFLICT-FREE FOR ALL RECEIVERS BY ITS TRANSMITTER IF NO CONFLICTING TRANSMISSIONS ARE DETECTED FOR A PERIOD OF 2D SECONDS AFTER THE START OF TRANSMISSION.

FACT: A TRANSMISSION MAY BE CONFLICT-FREE WITH RESPECT TO ITS INTENDED RECEIVER EVEN IF AN OTHER TRANSMISSION IS DETECTED BEFORE THE 2D SAFETY PERIOD.



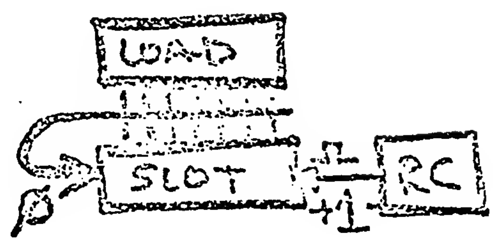
EROX

ETHER BARGAINING LOGIC

WE PRESUME WE KNOW THE ETHER'S DIAMETER AND THAT IT IS SMALL. WE PROPOSE THE FOLLOWING LOGIC FOR A STATION'S BARGAINING WITH THE ETHER.

FIRST, A CLOCK; CALL IT THE ROUND-TRIP CLOCK (RC). THE RC NEED NOT BE VERY GOOD; AN UGLY MULTI-VIBRATOR PERHAPS. IT SHOULD HAVE A PERIOD OF 2D+EPSILON, FOR SOME SMALL EPSILON.

SECOND, A COUNTER; CALL IT THE SLOT COUNTER (SC). THE SC IS ALWAYS COUNTING UP, INCREMENTED BY THE ROUND-TRIP CLOCK.

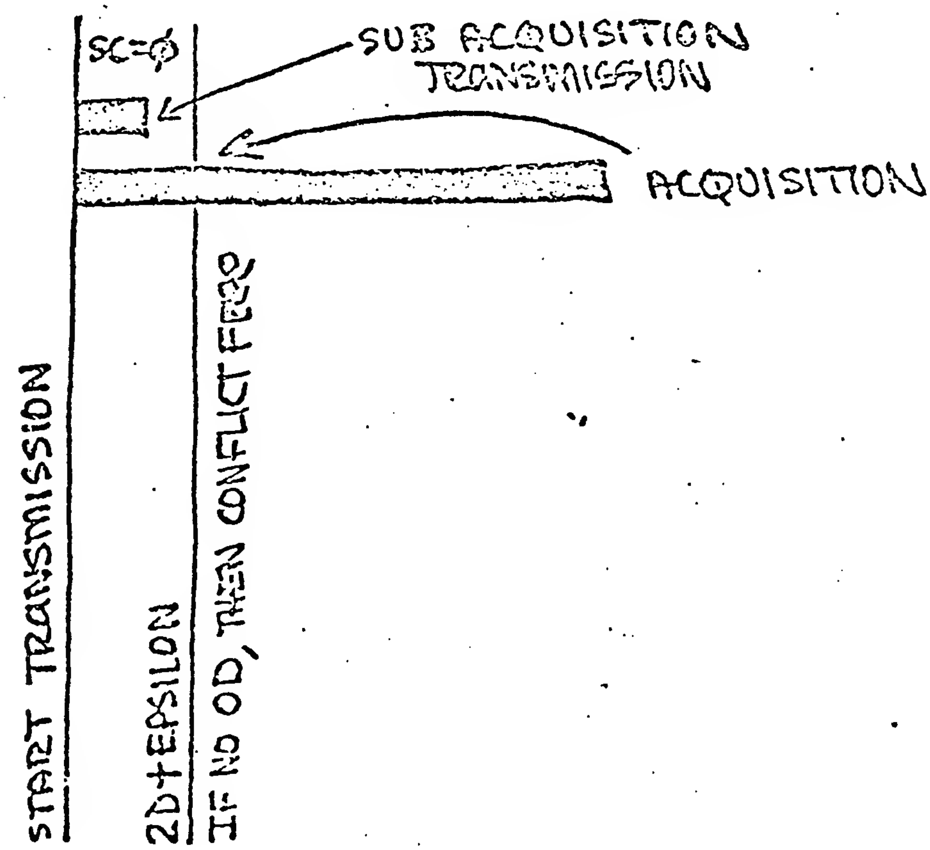


THIRD, A REGISTER; CALL IT THE LOAD REGISTER (LR). THE LOAD REGISTER TELLS THE SLOT COUNTER WHEN TO RETURN TO ZERO. THE LR HOLDS A NUMBER WHICH IS A MEASURE OF ETHER TRAFFIC LOAD. IN COUNTING UP FROM ZERO, THE SLOT COUNTER RETURNS TO ZERO WHEN ITS CONTENTS ARE EQUAL TO THAT OF THE LOAD REGISTER. THE LOAD REGISTER DEFINES THE LENGTH OF THE SLOT COUNTERS CYCLE.

FOURTH, OTHER-DRIVE DETECTOR, OD. THE OD LOOKS AT THE ETHER TO DETECT WHEN THE ETHER IS BEING DRIVEN BY SOME TRANSMITTER OTHER THAN ITS OWN, AT THE POINT OF THE TRANSMITTER.

FIFTH, THE OTHER-DRIVE DETECT BIT; ODB. THIS FLIP-FLOP IS SET WHENEVER THE OTHER-DRIVE DETECTOR DETECTS SOME OTHER TRANSMITTER'S DRIVE ON THE ETHER. BY THE DEFERENCE AXIOM, THE SETTING OF THE ODB CAUSES ANY TRANSMISSION IN PROGRESS TO BE IMMEDIATELY ABORTED. (THE ODB IS CLEARED WITH EACH TICK OF THE ROUND-TRIP CLOCK.)

SIXTH, THE NO-CONFLICT BIT, NCB. THIS FLIP-FLOP IS SET WITH THE FIRST BIT OF A TRANSMISSION ONTO THE ETHER BY THE LOCAL TRANSMITTER. THIS BIT IS CLEARED BY THE OTHER-DRIVE DETECTOR, ONLY DURING THE FIRST ROUND-TRIP OF A TRANSMISSION -- ONLY WHILE THE SLOT COUNTER IS ZERO.



XEROX

WHEN A STATION DESIRES TO TRANSMIT, IT WAITS UNTIL THE ETHER IS EMPTY AND THE SLOT COUNTER IS ZERO. IT THEN BEGINS TRANSMISSION, THE PLACING OF BITS INTO THE ETHER.

IF THE OTHER-DRIVE BIT COMES ON BEFORE END-OF-TRANSMISSION, THEN THE TRANSMISSION IS ABORTED -- THE DEFERENCE AXIOM. (WE MIGHT RECONSIDER THIS POSITION -- THE CONFLICTING TRANSMISSION MAY BE GOING IN THE OTHER DIRECTION). ?

AT THE START OF ACTUAL TRANSMISSION, THE NO-CONFLICT BIT IS SET. IF THAT BIT IS SET AT THE FIRST TICK OF THE ROUND-TRIP CLOCK, THEN A CONFLICT-FREE TRANSMISSION HAS OCCURRED. THIS EVENT MAY BE SIGNALED DURING TRANSMISSION IF THE TRANSMISSION IS LONGER THAN 2D SECONDS, OR AFTER THE END OF TRANSMISSION, IF THE TRANSMISSION IS LESS THAN 2D LONG. THUS THE STATION CAN KNOW TO SOME HIGH PROBABILITY THAT ITS TRANSMISSION HAS SUCCEEDED. (DISREGARDING NOISE)

XEROX

Xerox Sensitive

THE SLOT COUNTER HAS THE FOLLOWING PURPOSE. AS AN ONGOING TRANSMISSION COMES TO AN END, ALL THE WAITING STATIONS WILL WANT TO JUMP ON WITH THEIR TRANSMISSIONS -- AND THESE WILL OFTEN CONFLICT -- MORE OFTEN WITH LOAD. THE SLOT COUNTERS IN THE VARIOUS STATIONS WILL TEND NOT TO BE SYNCHRONIZED SO THAT THE SLOT COUNTERS WILL HOLD OFF SOME OF THE STATIONS GIVING (HOPEFULLY) ONE OF THEM TIME TO ACQUIRE THE ETHER. (OR JUST USE IT) FOR SHORT TRANSMISSIONS, ACQUISITION WILL NOT OCCUR AND THE ETHER WILL EXPERIENCE RAPID TRANSMISSIONS, HOPEFULLY ONE PER "SLOT". FOR LONG TRANSMISSIONS, THE FIRST STATION TO THE ETHER WILL ACQUIRE IT, THUS QUEUEING UP THE OTHER STATIONS TO WAIT THEIR TURN.

IN THE EVENT THAT A CONFLICT IS DETECTED, THE STATION HAS TWO OPTIONS. FIRST, IT CAN CLOBBER ITS SLOT COUNTER TO RANDOMLY MOVE IT AROUND IN THE QUEUEING CYCLE; AFTER A WHILE THE TERMINALS SHOULD BECOME DISTRIBUTED OVER THE VARIOUS SLOTS OF THE LOAD CYCLE. OR, THE STATION MIGHT CHOOSE TO, IN ADDITION, INCREMENT THE CONTENTS OF THE LOAD REGISTER, TO REDUCE ITS LOAD ON THE ETHER. AS THE ETHER BECOMES MORE LOADED WITH TRAFFIC, ALL OF THE STATIONS WILL THEREFORE BACK OFF TO SHARE THE ETHER 'OPTIMALLY'. OF COURSE, WITH SUCCESS ON THE ETHER, STATIONS MUST CONSIDER REDUCING THE CONTENTS OF THE LOAD REGISTER, TO TIGHTEN UP IN THE FACE OF REDUCED TRAFFIC. A STATION CAN BEGIN TRANSMISSION IN A SLOT WITH PROBABILITY $\frac{1}{\text{LOAD}}$.

NOT FOR PUBLIC CONSUMPTION

TO: INWG SUBGROUP AT STANFORD

FROM: Bob Metcalfe At Xerox PARC

Subject: A MODULAR VIEW OF THE INWG HOST-HOST PROTOCOL

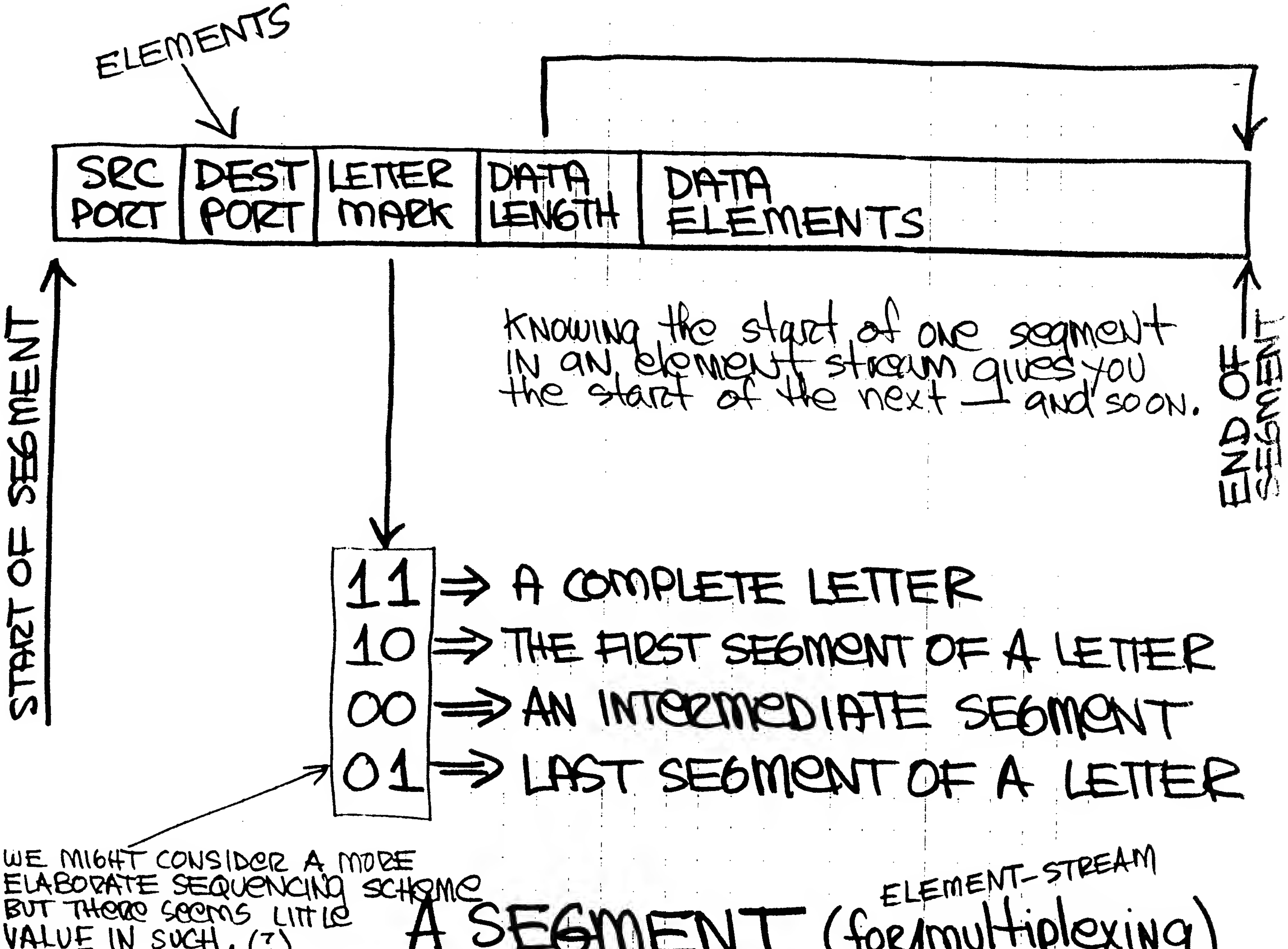
DATE: 17-JULY-73

THESE ARE MY <u>ROUGH</u> NOTES ON THE SORTING OUT OF WHAT SHOULD BE DONE WHERE. (AS PER YESTERDAY'S MEETING) BASIS FOR DISCUSSION. I introduce the notion of "SEGMENT". I try to make "whose multiplexing what" explicit. I try to keep various modules doing only what they need to do: modularity. Through all of this I gain the generality of packing small "Letters" into single messages for more efficiency.

A "LETTER" IS A SEQUENCE OF ELEMENTS OF SPECIFIED LENGTH, SOURCE, AND DESTINATION.

AT THEIR SOURCE, <u>LETTERS</u> ARE MADE INTO "SEGMENTS". <u>SEGMENTS</u> FROM A GIVEN LETTER ARE MERGED INTO THE <u>ELEMENT STREAM</u> GOING TO THEIR DESTINATION HOST. WHILE SEGMENTS OF VARIOUS LETTERS ARE MIXED FREELY IN THE OUTGOING ELEMENT STREAM, SEGMENTS OF THE SAME LETTER ARE SENT IN ORDER SO AS TO BE DIRECTLY ASSEMBLABLE INTO LETTERS AT THE DESTINATION HOST. A SEGMENT NEED ONLY CARRY THE SOURCE AND DESTINATION PORT IDENTIFICATION BECAUSE OTHER ADDRESS INFORMATION IS CARRIED FOR ALL SEGMENTS IN THE INTERNATIONAL MESSAGE HEADER.

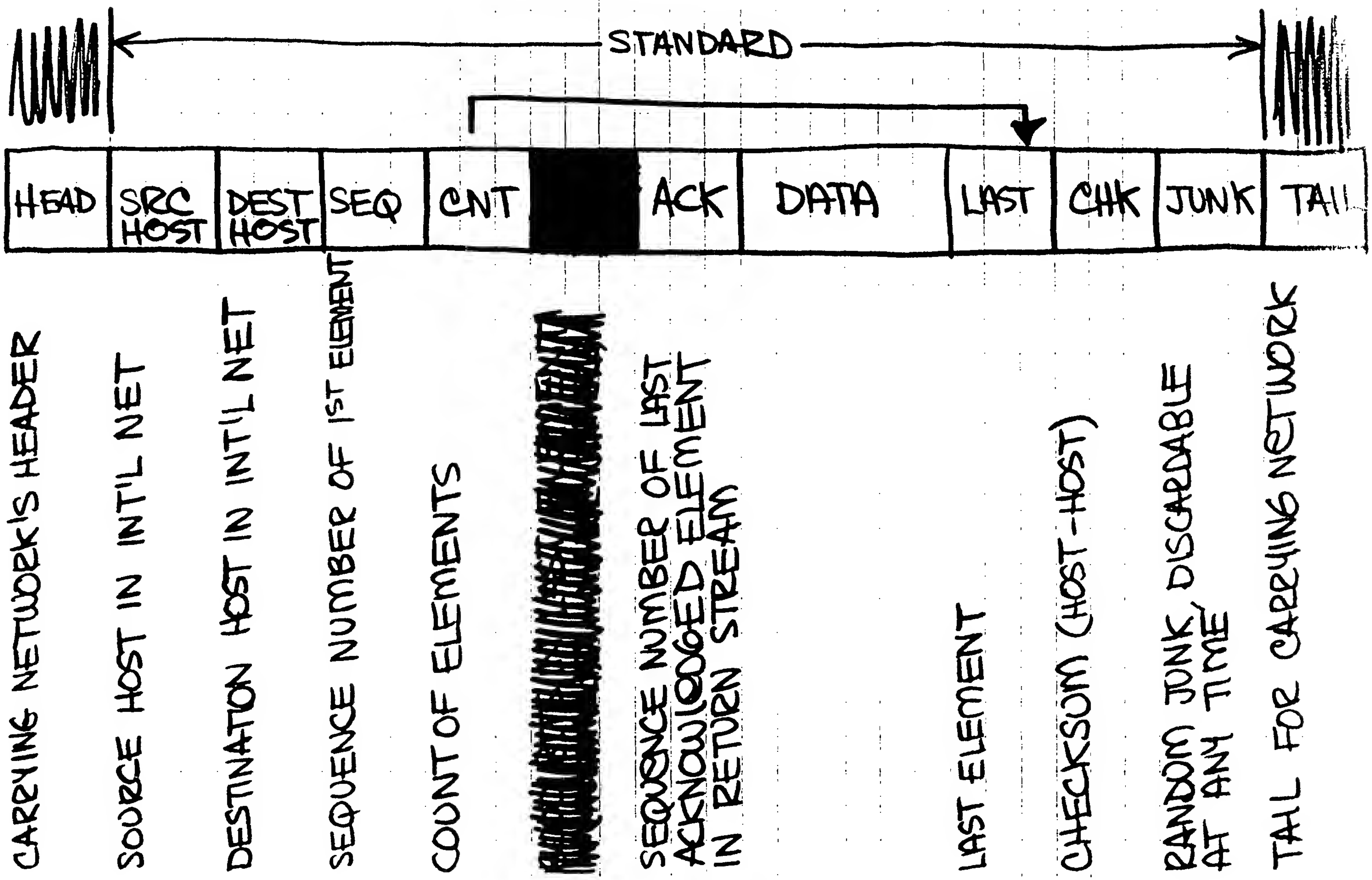
A "SEGMENT" IS A SEQUENCE OF ELEMENTS OF SPECIFIED LENGTH, SOURCE PORT, DESTINATION PORT, LETTER MARK. A SEGMENT'S SOURCE AND DESTINATION HOST ARE KNOWN, BUT NOT CONTAINED IN IT; THEY ARE IMPLICIT IN THE PARTICULAR HOST-HOST ELEMENT STREAM WITH WHICH THEY ARE ASSOCIATED. SEGMENTS ARE MERGED INTO AND EXTRACTED FROM THE APPROPRIATE HOST-HOST ELEMENT STREAM AS A UNIT. THE SIZE OF A SEGMENT IS BOUNDED ABOVE BY THAT OF ITS LETTER OR THAT BEYOND WHICH THE ONE LETTER WOULD BE GETTING TOO BIG A PIECE OF THE HOST-HOST ELEMENT STREAM ALL AT ONCE. SEGMENTS ARE SIZED SO AS TO FAIRLY SCHEDULE TRANSMISSIONS, TO KEEP LATENCY DOWN, LIKE QUANTA IN A TIME-SHARING SYSTEM.



A SEGMENT (for element-stream multiplexing)

EXISTS ONLY IN ASSOCIATION WITH A HOST-HOST ELEMENT STREAM

AS SEGMENTS ARE DROPPED INTO AN ELEMENT STREAM, THEIR INTERNAL STRUCTURE BECOMES <u>INVISIBLE</u> UNTIL IT IS NECESSARY TO RECONSTRUCT THEIR LETTERS AT THE DESTINATION HOST. HOST-HOST <u>MESSAGES</u> ARE CONSTRUCTED FROM THE RAW ELEMENT STREAM WITHOUT REGARD FOR THE ELEMENTS THEMSELVES AND, IN PARTICULAR, NO KNOWLEDGE OF SEGMENT OR LETTER BOUNDARIES. MESSAGE SIZES ARE CHOSEN BY THE MANAGER OF THE HOST-IMP "PORT" TO OBEY THE RULES OF HOST-IMP PROTOCOL AND, ALSO, SO AS TO FAIRLY SHARE THE HOST-IMP PORT AMONG THE VARIOUS, COMPETING HOST-HOST ELEMENT STREAMS.



A MESSAGE (for multiplexing the international network)

A GATEWAY DEALS ONLY IN MESSAGES. ITS PRIMARY TASK IS TO EXTRACT THE STANDARD MESSAGE OUT OF THE MESSAGES FROM ITS VARIOUS CARRYING NETWORKS AND TO CONSTRUCT FROM THE STANDARD MESSAGE, A MESSAGE OR SEVERAL MESSAGES APPROPRIATE FOR THE NEXT CARRYING NETWORK. IN PRINCIPLE, A GATEWAY COULD MAKE BIG MESSAGES OUT OF SMALL ONES, BUT IT WOULD NEED TO MATCH HEADERS AND LOOK FOR CONTIGUOUS ELEMENT SEQUENCES; SO NO GO.

TO MAKE SEVERAL STANDARD MESSAGES OUT OF ONE, THE GATEWAY WOULD DUPLICATE THE FOLLOWING FIELDS IN EACH SMALLER MESSAGE:

(1) SRC HOST
(2) DEST HOST
(3) ACK ELEMENT SEQUENCE

THEN THE DATA WOULD BE DISTRIBUTED WITH THE APPROPRIATE COUNT AND SEQUENCE NUMBER OF THE LEADING ELEMENT. THE CHECKSUM WOULD THEN (OPTIONALLY) BE COMPUTED FOR EACH SMALL MESSAGE. WE ASSUME THE CHECKSUM WAS CORRECT ON INPUT.

STANDARD MESSAGES, ENCAPSULATED IN MESSAGES OF THE ADJOINING CARRYING NETWORK, WOULD BE EXTRACTED AND SORTED ACCORDING TO SOURCE HOST. MESSAGES ARRIVING WITH BAD CHECKSUMS SHOULD BE LOGGED AND DISCARDED IMMEDIATELY UPON RECEIPT.

WE NOW HAVE MESSAGES ARRIVING TO THE RECEIVING END OF A HOST-HOST ELEMENT STREAM. THE ELEMENT STREAM IS RECONSTRUCTED IN THE FAMILIAR WAY. SEGMENTS AND LETTERS ARE IRRELEVANT TO SUCH RECONSTRUCTION. WE USE THE STANDARD WINDOW SYSTEM AS PER "VIRTUAL PATH".

AN "ERROR-free" HOST-HOST ELEMENT STREAM IS NOW Available AS INPUT TO THE PROCESS OF RECONSTRUCTING LETTERS. SEGMENT AFTER SEGMENT IS PULLED OUT OF THE ELEMENT STREAM AND ROUTED TO THE INDICATED DESTINATION PORT. As LETTERS ARE collected from CONSECUTIVE SEGMENTS, PROCESSES ARE NOTIFIED.

POINT: MANY SEGMENTS, EVEN MANY LETTERS, CAN NOW BE COLLECTED INTO A SINGLE HOST-HOST TRANSMISSION. LETTER AND SEGMENT BOUNDARIES ARE NOT VISIBLE TO THE MODULE WHICH CONSTRUCTS messages FROM THE VARIOUS HOST-HOST ELEMENT STREAMS.

RESORCE ALLOCATION IS IMPROVED BY ADDING FREEDOM TO THE MULTIPLEXING OF A HOST-HOST STREAM INDEPENDANT OF THE MULTIPLEXING OF THE HOST-IMP PORT.

D R A F T   P A R C   C S L   M E M O R A N D U M

TO: Boggs, Deutsch, Duvall, Fiala, Lampson, Liddle, McCreight
Rider, Simonyi, Sproull, Sturgis, Taft, Thacker, Zelinsky
and others who may wish to find themselves involved

FROM: Bob Metcalfe

SUBJECT: A Proposed Pup -- Parc Universal Packet

DATE: March 19, 1974

DRAFT
COMMENTS PLEASE

This memo is written and should be read with caution; its purpose is to promote a standard. Because there isn't an ice cube's chance in hell that our (or anyone else's) standard will be adopted without interminable debate and revision, the memo itself is quick and dirty. This way we got the ball rolling early. For once, our style is of no interest.

Successful implementation of the standard will require a multi-lateral agreement among the czars of Parc's various existing and planned packet networks. The instrument of this proposed agreement is to be an object we affectionately call a "Pup", a Parc Universal Packet.

A list of the packet networks at Parc would include, in arbitrary order of pedigree, (1) Ethernets, (2) Localnets, (3) Arpanets, (4) MCAnets, and (5) EIAnets. All have been considered, more or less, in the personal turmoil leading to our current Pup proposal.

A problem barks for our attention: How do we intelligently interconnect these networks and the computers to which they are attached?

We propose that a standard packet protocol be adopted to allow processes living on any of our interconnected computers to send packets among themselves through any of our interconnected networks. Adoption of such a standard would give us a general interprocess communication system. Not all host-host or process-process communications need use the general Pup system, of course; keeping this in mind will help us prevent our Pup from becoming a real Dog ("Disgustingly Over General"). But, those that do use Pups won't care which of our various networks and computers are involved. And this can be a good deal. Arguments?

Imagine the economies of being able to access the following resources from any process in the known interconnected world: (1) Rider's Slot printing server, (2) Boggs's magtape-controlling Altos, (3) the terminals coming in from the DLS machines, (4) Deutsch's Arpanet NCP on Polos, (5) the Polos editors and formatters, (6) the Parc file system (Maxc for now), and a HOST of others; the NET result is IMPortant, no matter how you PACKET.

Perhaps this memo addresses the "Protocols" issue which Butler pointed to during his distributed file system Dealer; maybe it steps toward the "Reliability" methodology also listed.

[1] Principles.

<1a> Heterogeneity. We recognize that Parc has its various networks, not principally because of any lingering attachment to obsolete capital equipment, but presumably because each network seems more suited to

RELIABILITY
is not having to say you're sorry.

certain applications than the others. Therefore, in interconnecting these networks and their computers, we must not simply wash out the differences with emasculating standards, but breed the benefits of underlying variety.

This is to say that we do not intend the proposed standard to be all things to all persons; many will and should avoid internetwork standards to get at the particular capabilities offered by their favorite network.

<1b> Encapsulation. Because we are to retain the advantages of our differing networks, Pups -- packets carried according to our internetwork standard -- must be a subset of the packets carried by each network. Thus, a Pup is to be encapsulated as it passes through any one network so as to be a recognized type of packet in that network. As only one of many types, the General Pup standard will invoke (burdensome?) processing overhead only on those packets requesting it.

<1c> Gateways. The active components in the system to be built according to the standard protocol are (to use internetworking jargon) as "Gateways", processes which take packets from one network, diddle them, and hand them to another. From our point of view, the processes which we often call Network Control Programs (NCPs) are already Gateways; they transmit packets from the software networks of processes inside computers to the hardware networks outside. With internetwork connections we introduce additional Gateways; processes which actually sit at the junction of two (or more) hardware networks.

Gateways can be seen as store-and-forward packet-switching nodes at the internetwork level; they are like meta-Imps. In a host connected to several networks it is likely there would be several Gateways (i.e., NCPs), one for each, and yet another putting them all together for internetwork switching.

<1d> Best Efforts. We believe that when a packet is given to a raw process-process packet system, the system should promise to give only its best efforts to deliverence; the sender of the packet should expect its successful delivery only with some stated high probability (say 9/10, or 99/100, or 999/1000) conditional on the destination being ready and waiting. Higher quality communication, say with additional error control and some flow control, should be provided with algorithms implemented at the application-process level.

This idea comes closest to a methodology for building reliability into distributed systems -- thin-wire best-efforts communication among processes. (see thesis)

[2] The Parc Universal Packet

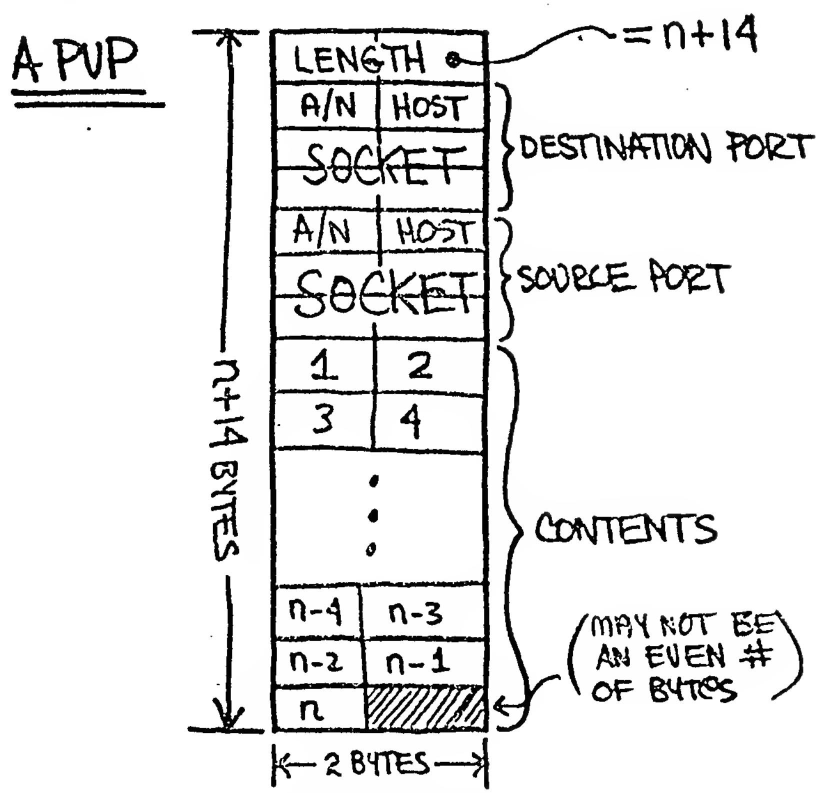
The Pup is a collection of 8-bit bytes. The first 2 bytes of a Pup carry its length. Next come two port addresses, each 6 bytes. And then come its content bytes about which the Pup definition has nothing to say. When a process hands a Pup to its local Gateway, it expects that the packet will be delivered to the waiting destination only with some high, yet to be specified (say 99/100), probability. That's it.

A Pup is a virtual packet; virtual to the source and destination processes involved. On the wires or in the memory of some specific network or computer it may look different, encapsulated, with fields rearranged, with extra fields containing network-specific data. You'll see what we mean in the example of how a Pup might be transported through an Ethernet, below.

<2a> Length. We propose that the length of a Pup be carried as the number of 8-bit bytes in the Pup, including the length itself, the addresses, and the contents. (We waver between 8-bit bytes and bits; but now choose the former.) The purpose of this length is to facilitate transport of the Pup without disrupting its content through the inadvertent adding or subtracting of bits. In fact, as a Pup winds its way through various networks, it is likely that it will acquire a tail, extraneous trailing bits ("padding"), but these are OK; they can be routinely ignored.

<2b> Addresses. A Pup has two addresses. The first is that of the destination port; the second is that of the source port. A port address is 6 bytes (48 bits) long and identifies an area/network, a host computer, and a "socket" in that host. We use the first 8 bits to identify the area/network of a port. We use the second 8 bits to identify the port's host computer in the identified area/network. We use the remaining 32 bits to identify the port's socket within the identified host. This addressing scheme was bred for generality from those of the Localnet, the MCAnet, and the Arpanet. (Its seems that no harm is done to Pups if a host on more than one network is known by several names, one for each of those networks. Thought.)

<2c> Contents. The definition of a Pup is silent about the contents of a Pup. So-called higher-level protocols, like the Byte-Stream Protocol (BSP) to follow below, go further by putting meaning on certain Pup contents.



<2d> Pup Problems. There are (at least) three problems which need further study. One is that of the largest Pup to be permitted. Another

is that of finding appropriate time-outs. And a third is that of undeliverable packets in a multi-path network of Gateways.

The Localnet carries small fixed-length packets of 512 bits, some of which are used for control information. It may be we need to adopt a maximum size for Pups so they'll fit in a Localnet's packets; this would spell doom for raw packet efficiency, especially for the lower performance networks where it counts. (512-(80+112+48))/512 is less than 54%, right off the top. We might invent a hairy scheme for fragmenting Pups while in a Localnet. Or, and this seems the most attractive right now, we could stick those communicating processes looking for high bandwidth with the job of deducing the largest Pups they can use, with brute-force trial-and-error. Suggestions?

80 ← LOCALNET
112 ← PUP
48 ← BSP(+)

YES!

Time-outs are central to our brand of reliable communication. If time-outs are chosen too small, our networks become cluttered with duplicate, successfully delivered packets. If time-outs are chosen too large, our networks sit idle, with everyone twiddling their thumbs. Choosing good values for time-outs is complicated somewhat when the transporting mechanisms are unknown, as for Pups. A Pup may just take a hop through a 50 Mbps circuit in a directly connected Localnet, or through an earth-orbiting satellite in the Arapnet, to use the extremes. We suggest that our programs, to the extent they want to improve the efficiency of their use of the Pup system, attempt to zero in on optimal time-outs (with round-trip time measurement and time-out adjustment) in the course of their various communications; a mild form of the trial-and-error approach, again.

YES!

If there were to exist more than one path from Gateway to Gateway toward a single destination, then it would be possible that Pups intended for an unreachable destination would bounce back and forth between well-meaning but uninquiring Gateways. This problem could be solved by not having alternative routing at the Gateway level; this is the easy way. We might also provide for shared routing information among Gateways like that shared among Imps in the Arpanet. Or we might put a Gateway handling count (hop count) in each Pup so that, when it's been around for too long, it gets discarded. The first of these solutions seems the best, for right now; the last seems better in the longer term. Suggestions?

<2e> Pup Gateway Routing. A process instructs its operating system to transmit a Pup. The Pup is bundled up in the local Gateway, often called the NCP, and its destination address examined. If the area/network field of the destination port is seen to match that of a locally connected network, then the Pup is appropriately encapsulated and sent to the indicated destination host. If the area/network field of the destination port does not match that of a locally connected network, the Gateway must consult its own routing table to discover how to get the Pup on toward its destination, to discover which host on a directly connected network is in a position to take the Pup into another area/network closer to the destination. Gateway routing tables need only contain a number of entries equal to the number of existing networks, not one for each host or (ugh) process.

[3] Ethernet Transport of a Pup

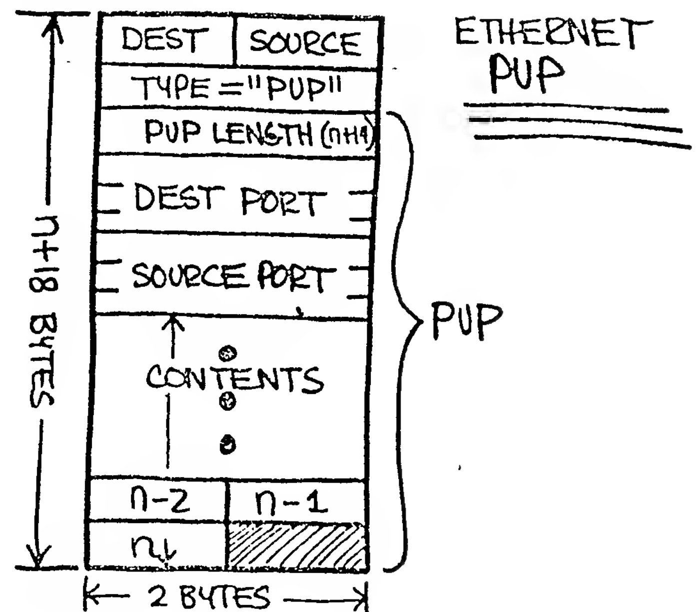
By way of example, here is how we propose to carry a Pup in the Ethernet.

First will come the required 16 bits of Ethernet addressing data. A destination and source must be specified in each Ethernet packet, 8 bits for each. The source field is fixed by the Gateway's host address on the transporting Ethernet. The destination field is to be derived from the Pup's 48 bits of destination address; it might be the specified host number if the area/network matches that of the current transporting Ethernet, or it might be the Ethernet address of a host willing to forward the Pup onward to its intended destination elsewhere. = GATEWAY

Next comes the Ethernet's own standard type word with the specific type "Pup". This type is used by the receiving Ethernet's host's NCP to recognize the packet as a Pup and to give it the Gateway handling it deserves. Such handling might be (1) nothing more than handing it off to the addressed port's process, or (2) perhaps routing and reformatting for injection into an MCAnet, or Localnet, or Arpanetnet, or (3) perhaps immediate discard for any one of a number of reasons.

There are several reasons why a packet might be discarded and lost at a Gateway: (1) nobody here by that name, (2) Pup too big to fit in transporting packet, (3) congestion too high right now, (4) can't get there from here, or (5) unrecoverable transmission error. The source of a packet can't, no, shouldn't, care which of these strikes his packet down in its prime; the recovery procedure is the same, just time-out and retransmit.

And next would come the Pup itself, in its entirety, transported in the required number of 16-bit words.



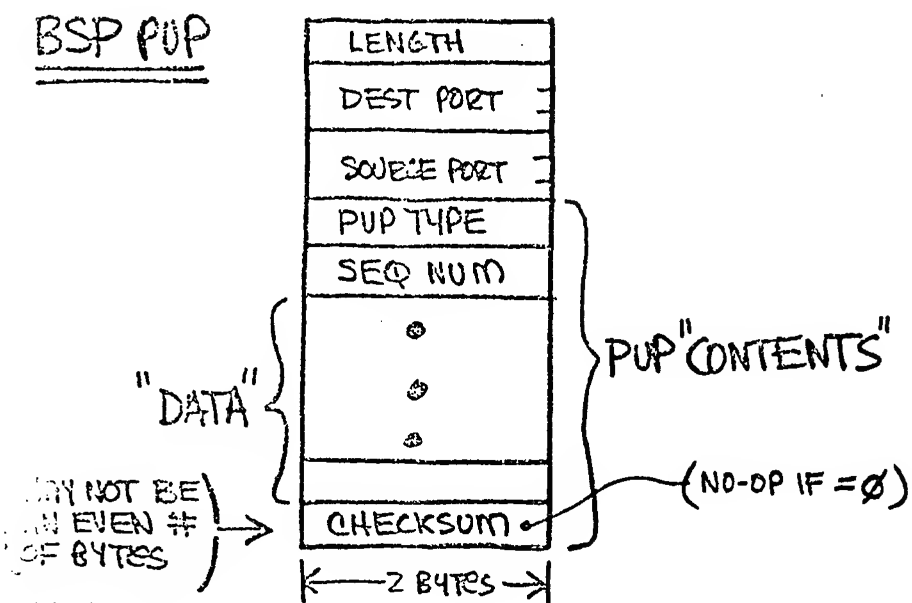
[4] Byte-Stream Protocol

And now, as an example, we offer a protocol with which two processes would transfer an error-free, flow-controlled byte-stream at the maximum rate possible, using Pups.

We now specify that the first 2 "content" bytes of Pups being used by processes wishing to use the Byte-Stream Protocol be used to carry a Pup type. A registry of such types should be kept. We define the use of

the following Pup types: RTS, STR, Data, End, Abort, Ceck, Sack, and Nak.

We propose that each of these types carry as Pup "contents" a 2-byte sequence number and a trailing (optional) 2-byte software checksum word (two bytes inside their Pup; if non-zero, then its an add-and-cycle over the Pup's contents minus checksum.)



<4a> Connecting Up.

First, some distinctions. (1) Each Pup has a source and a destination. (2) Each byte-stream connection has a user and a server, often called an

initiator and a listener. (3) Each byte-stream has a sender and a receiver. Those are orthogonal descriptors; a process may be one or the other of the above pairs, independently.

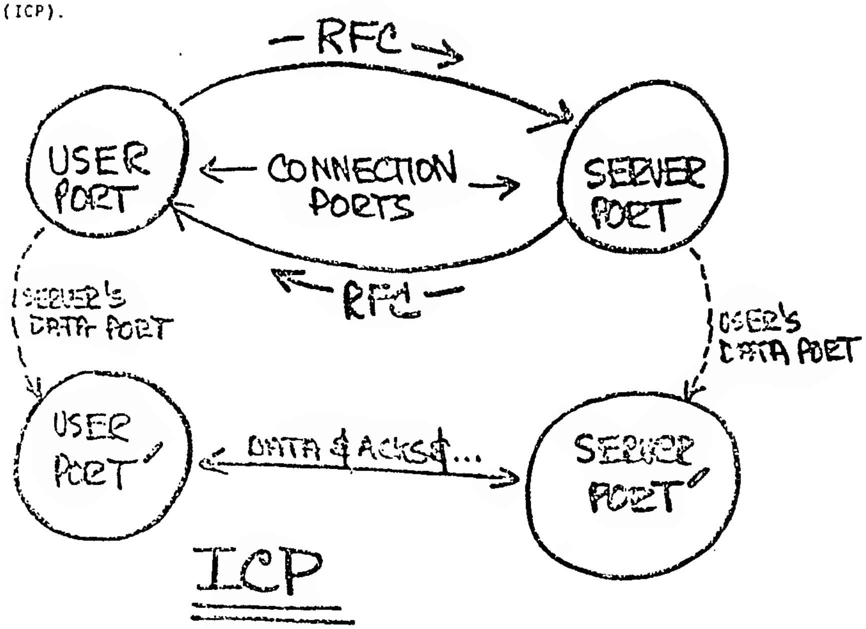
We are given two processes wishing to transmit a byte-stream from one to the other. A process port into one of these processes, the server or listener, is known to the other, the user or initiator.

The first action taken is by the server who registers with his local Gateway that he wishes to be given Pups addressed to a specified publically-known port.

The next action is taken by the user who registers 1 or 2 ports of his own with his local Gateway (NCP). He sends a request for connection (RFC) in a Pup from his connection port to the known server connection port. The RFC is either an STR (sender to receiver) or RTS (receiver to sender), depending on whether the user wants the byte stream to move to or from him. The RFC carries in it the user's byte-stream port (possibly not the RFC's source) and a byte-stream name. (The byte-stream name might be used as a file name by a file transfer server.)

Upon getting the user's RFC, the server decides to serve the byte-stream transfer, establishes a (possibly) new port for the transfer, and sends a matching RFC (STR for RTS or RTS for STR, with copied sequence number and server-assigned name) to the user's RFC's source port. When the answering RFC is received by the user, the byte-stream connection has been initiated and all exchanges afterward happen between the user and

server byte-stream ports. This simple exchange is a packet version of the Arpanet's lumbering connection-version Initial Connection Protocol (ICP).



<4b> Data Packets. Bytes in the byte-stream are carried from stream sender to stream receiver in Pups of type "Data". Following the Pup type is a 16-bit byte sequence number, starting at zero, identifying the first byte of the Pup as the 0th, 1st, 2nd, ..., or ith byte of the stream, with wrap-around on 16 bits. Then come the bytes themselves. Then comes the (optional) 16-bit checksum. That's it.

<4c> End Packet. The End packet is launched by the sender to indicate

where the end of stream is. The 16-bit byte sequence number in the End packet is that of the first byte after the end of stream. And then, the End's checksum.

<4d> Abort Packet. An Abort packet can be sent by either the stream sender or stream receiver at any time. Abort messages can be completely ignored, but everything will work more smoothly if they cause the stream transfer to stop immediately. Each Abort packet will carry a 16-bit code (from a registry of codes for program processing) and a text string suitable for human consumption. (Remember the Arapanet FTP?) And then would come the checksum, as usual.

<4e> Cack Packet. A Cack is a "cumulative ack". It travels from the stream's receiver to the sender indicating that, since the beginning of the stream, all bytes through the indicated sequence number were received correctly. Also, as for the following Sack and Nak, the Cack carries an allocation for flow control.

An allocation is a 3-tuple recommending (1) the number of bytes per Data Pup which the receiver is prepared to receive, (2) the number of Data Pups which the receiver is prepared to receive, and (3) the number of stream bytes the receiver is prepared to receive, all as of the time of departure of the Cack. And then the standard checksum.

The purpose of the allocations carried from receiver to sender is to allow the sender to participate in the optimization of byte transfer. It should be remembered that the network itself may impose further

restrictions (say maximum transportable Pup size), not to mention those from the sender.

[illegible] Sack Packet. A Sack is a "specific ack". It reports the successful arrival of the specified number of bytes at the indicated position in the byte stream. The Sack is a completely redundant message which can be sent by a helpful receiver to indicate the arrival of a Data packet carrying data past a hole in the stream; its purpose is to cut down on retransmissions. A sender who gets ahead on his transmissions -- has several packets outstanding at a time -- can use Sacks to avoid retransmitting certain sections of stream. Completely redundant; but maybe helps performance.

[illegible] Nak Packet. A Nak is a "negative acknowledgement". The Nak is a completely redundant packet which can be sent by a helpful receiver to indicate the known loss of a specified number of bytes at the indicated position in the byte stream; its purpose is to hasten the retransmission of data lost at the receiver due to congestion, lack of space, or the like.

THESE FEW PACKETS ALLOW A RANGE OF COMPATIBLE TRANSMISSION ALGORITHMS. OBVIOUS?

[illegible] Next?

[illegible] might now like to see other written contributions? Perhaps we could have a meeting for development, discussion, and revision of this proposal? Soon? Comments?